From the desk of
Günther Dorth

no p. 11 re-print, App A p 1, 2, 3, ~~5~~ 6, ~~7~~

I thought you might be interested in this

GD

send him page 11
App 1, 2, 3, 6.

Apparently earlier than what I have.

Missing p. 11
A 1
A 2
A 3
A 6

WORKING PAPER

BALLY BASIC HACKER'S GUIDE

NOTE: The following pages are rough copy which have not been edited and not intended for publication.

cat

7/25/79

BALLY BASIC HACKER'S GUIDE

This report describes features provided in the Bally Basic Cassette, but not documented in the programmed instruction course booklet.

Some of these features may be removed to make way for other improvements

ABS FUNCTION

The absolute value function is available. It is typed in as 3 discrete keystrokes: "A", "B", and "S".

STOP COMMAND

The STOP command halts the program. It is typed in as 4 discrete keystrokes.

RM

The special variable RM has its value the remainder produced by the most recently executed integer division operation.

This example prints M mod 256

```
10 A = M ÷ 256
20 PRINT RM
```

XY

This variable remembers the X, Y position specified in the latest LINE command. The Y value occupies the high order byte of this word; X the lower byte.

CALL N

The CALL command transfers control to an assembly language subroutine. This routine should terminate by executing a RET instruction. Register DE contains a pointer to the line being executed. If needed it should be saved then restored before returning to Bally Basic.

Example:

```
500 CALL 0      (Self Destruct Reset)
```

:RUN Command

Used to load a 128 byte boostrap routine from the Bally Basic Cassette Tape Interface. This bootstrap loads into screen memory beginning at address $4ØØØ_{16}$. Execution begins there when the load is completed. This feature was provided to allow short assembly language programs to be loaded from tape.

A transfer vector to the get character from tape routine is located at $2Ø1A_{16}$. The character comes back in A.

BALLY BASIC provides special commands for interfacing to the resident calculator program. These commands allow the addition, subtraction, multiplication and division of 16 digit numbers with eight digits on either side of the decimal point.

NUMBER REPRESENTATION

The @() array is used to store the numbers operated on by these commands. Eighteen consecutive @ elements are used, one per digit. (The two extra digits are for sign and overflow indicators). The assignment of digit positions to array elements looks like this:

RELATIVE DIGIT POSITIONS		
Ø	digit	10^{-8}
1	digit	10^{-7}
2	digit	10^{-6}
3	digit	10^{-5}
4	digit	10^{-4}
5	digit	10^{-3}
6	digit	10^{-2}
7	digit	10^{-1}
8	digit	10^{0}
9	digit	10^{1}

RELATIVE DIGIT POSITIONS

1Ø	digit	10^2
11	digit	10^3
12	digit	10^4
13	digit	10^5
14	digit	10^6
15	digit	10^7
16	overflow indicator * (non-zero if overflowed)	
17	sign (Ø if positive, 8 if negative)	

*overflow indicator must be initialized to Ø on command entry.

The digits may be represented as binary numbers between Ø and 9 or as ASCII character codes between 48 and 57. The result is always ASCII.

This example sets up 3.14159 beginning at @(Ø).

```
1Ø FOR A = Ø to 18                    ; , clear everything
2Ø @(A) = Ø
3Ø Next A
4Ø @(8) = 3
5Ø @(7) = 1
6Ø @(6) = 4
7Ø @(5) = 1
8Ø @(4) = 5
9Ø @(3) = 9
```

COMMAND FORM

All four commands resemble this addition command example:

```
1ØØ$+ @(Ø), @(18), @(36)
```

which means "Add the number starting at @(Ø) to the number at @(18) and store the result beginning at @(36)".

<u>Examples of the others</u>

$-@(A), @(B), @(B)   subtraction

$×@(Ø), @(Ø), @(Ø)   multiplication

$÷@(X), @(K), @(N)   division

This example prints the sequence 1, 2, 4, 8, etc.

```
10 FOR A = 0 to 17; . INITIALIZE
20 @(A) = Ø
30 NEXT A
40 @(8) = 1 ; . start at 1
50 GOSUB 1000 ; . CALL PRINT ROUTINE
60 $ +@(Ø), @(Ø), @(Ø) ; , DOUBLE NUMBER
70 GOTO 50
80
1000 Z = 1 ; . SET LEADING ZERO FLAG
1010 IF @(17) = "8" PRINT "-",
1020 FOR B = 16 to 9 STEP - 1
1030 IF @(B) = "Ø" IF Z GOTO 1060
1040 Z = Ø
1050 TV = @(B)
1060 NEXT B
1070 TV = @(8)
1080 PRINT
1090 RETURN
```

IO PORT AND MEMORY ACCESS

The physical IO ports of the Arcade may be accessed through the &() construct.

&() is used much like @(). For example:

```
>&(23) = 255; &(21) = 255
```

sets ports 21 and 23 both equal to 255, which makes a rocket like sound.

Ports may be read by using &() in place of any expression. For example:

```
10 PRINT &(23)
20 GOTO 10
```

will loop sampling and reporting the status of the leftmost column of keys on your easy-entry keypad. Press any key in that column and see what happens. Try combinations.

The physical memory of the Arcade may be read or written in a similar way using %(). This example prints the first hundred words of operating system ROM:

```
10 FOR A = 0 to 198 STEP 2
20 PRINT %(A)
30 NEXT A
```

NEAT I/O PORTS

Color control ports

&(0) = COLOR 0 RIGHT VALUE

&(1) = COLOR 1 RIGHT VALUE

&(2) = COLOR 2 RIGHT VALUE

&(3) = COLOR 3 RIGHT VALUE

&(9) = HORIZONTAL COLOR BOUNDARY REGISTER

&(10) = VERTICAL BLANKING REGISTER

The format of these values is the same as the codes used with FC and BC (color × 8 + intensity).

These ports only have effect when the horizontal color boundary register is set to a value less than 44. This boundary register is set to the byte number of where to switch from one set of colors to the other.

The colors for the left side of this boundary are defined by FC and BC.

The following program demonstrates the ideas:

```
10 &(0) = 0
20 &(1) = 123
30 &(2) = 185
40 &(3) = 251
50 FOR A = 0 to 255
60 &(9) = A
70 NEXT A
80 GOTO 50
```

If you halt this program while the black background is up and study the screen, you can see how the program is stored intermixed with the graphics. Try adding lines to the program.

To hide the program but show graphics, set &(0) and &(1) to the background color; &(2) and &(3) to the foreground color.

The vertical blanking register, &(10) specifies how many scan lines of graphics data are to be displayed. All lines below the specified ones are shown in the background color. This register acts like a curtain which we can lower between acts. Its initial value is 176. Try 204 and watch Bally Basic's scratch storage area twinkle.

HAND CONTROLS

&(16)	Player 1	joystick & trigger
&(17)	Player 2	joystick & trigger
&(18)	Player 3	joystick & trigger
&(19)	Player 4	joystick & trigger

The value returned looks like this:

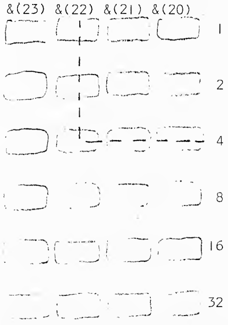
2^4	2^3	2^2	2^1	2^0
TRIGGER	RIGHT	LEFT	DOWN	UP
16	8	4	2	1

&(28)	Player 1	knob position
&(29)	Player 2	knob position
&(30)	Player 3	knob position
&(31)	Player 4	knob position

The value ranges from Ø (full clockwise) to 255 (full counterclockwise)

KEYPAD SENSING

While running a program you can check to see if a key has been pressed on the easy-entry keypad without going into wait.



For example to check for the 5 key on the easy-entry keypad overlay:

```
10 IF &(22) = 4 PRINT "5"
20 GOTO 10
```

LIGHT PEN INTERFACING

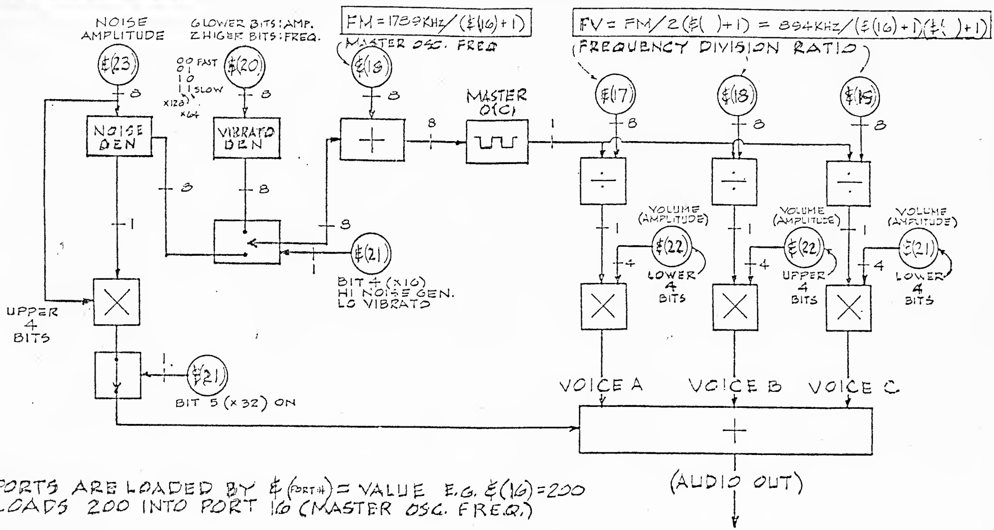
&(14)	Y	Coordinate position
&(15)	X	Coordinate position

These ports retain the coordinates of the most recent light pen hit.

Light pen sampling must be turned on. This is done by issuing the :INPUT command which is normally used by the cassette tape interface.

It is possible that the coordinates from &(14) and &(15) may be off by a constant amount due to time delays in the light pen circuits. You should add or subtract the proper fudge factor for your light pen to translate into screen coordinates.

BALLY BASIC SOUND SYNTHESIZER



PORTS ARE LOADED BY &(PORT#) = VALUE E.G. &(16)=200
LOADS 200 INTO PORT 16 (MASTER OSC. FREQ.)

PORT #

&(16) MASTER OSCILLATOR FREQUENCY
&(17) TONE A DIVISION RATIO CONTROLS FREQ. OF VOICE A
&(18) " B " " " " " B
&(19) " C " " " " " C
&(20) VIBRATO: UPPER 2 BITS FREQ., LOWER 6 BITS AMPLITUDE.
&(21) LOWER 4 BITS AMPLITUDE (VOLUME) OF VOICE C. BIT 4 HIGH SELECTS NOISE, LOW SELECTS VIBRATO TO CONTROL FREQ OF MASTER OSCILLATOR. BIT 5 HIGH SELECTS NOISE TO MIX WITH VOICES.
&(22) LOWER 4 BITS: VOLUME OF VOICE A, UPPER 4 BITS: VOLUME OF VOICE B.
&(23) ALL 8 BITS ARE AMPLITUDE OF NOISE FOR MASTER OSC. UPPER 4 BITS: VOLUME OF NOISE ADDED TO OUTPUT

THE BALLY COMPUTER SYSTEM SOUND SYNTHESIZER

The sound synthesizer can produce 3 tones at one time, with vibrato, noise, and amplitude control. Bally Basic normally uses only one of the tone voices. By using the &() construct, all of the features of the synthesizer can be accessed.

The synthesizer can be divided into two sections. The first section, on the left hand side of the block diagram, is concerned with controlling the master oscillator. The master oscillator output is input to the other section which contains the 3 voice oscillators. Thus changes made to the master oscillator side of the synthesizer effect all 3 voices.

The master oscillator is a programmable frequency divider. It is a counter which is clocked at 1.789 Mhz. Each time it counts down to zero, the state of its output is toggled and the counter is reinitialized to the last value output to port 16. The master oscillator output is a square wave of frequency 1789 Khz / (port 16 + 1)

By setting control bits in port 21 the behavior of the master oscillator may be modified. Bit 4 when set causes noise to be added to the value output to port 16. This sum is used to reset the counter in the master oscillator. The effect is that the frequency is varied by a random amount. The amount of variation is controlled by the noise amplitude port: &(23).

Reseting bit 4 turns off noise modulation, and turns on vibrato. Vibrato works like noise modulation, except that the value added to port 16 varies over a programmable range Ø-63 (vibrato amplitude) the rate at which this added value varies is determined by the vibrato frequency register, which can have four different values: Ø meaning fastest to 3 meaning slowest.

The right side of the synthesizer consists of three frequency dividers (voices) with associated volume control registers. Each divider is clocked

by the output of the master oscillator. The output frequency is given by the formula FV = FM 2(&(N)+1)

where N is 17 for voice A, 18 for voice B, or 19 for voice C. FM is the frequency output by the master oscillator. Substituting in the formula for FM we have:

FV = 894Khz/(&(16)+1)(&(N)+1)

Each voice has a 4 bit volume control register; Ø is quiet, 15 is full volume. The volume is linearly proportional to the value output. Unfortunately the volume control registers share physical port assignments with other control bits. Voice A uses the lower 4 bits of &(22); voice B the upper 4 bits. Voice C uses the lower 4 bits of &(21), which also deals with noise control. See "How to deal with shared IO ports" for explanation. While noise may be mixed in with the output of voices A, B, and C by setting bit 5 of &(21). The volume of this noise is determined by the upper four bits of &(23), the noise volume register.

HOW TO DEAL WITH SHARED IO PORTS

The Bally Home Computer System has several shared IO ports in its design. This means that several distinct registers are grouped into one IO port. For example: &(21) controls both the noise generator and the volume of Tone C. This design trick is a relic of the early days of microcomputers, when such shenanigans would save a TTL chip or two.

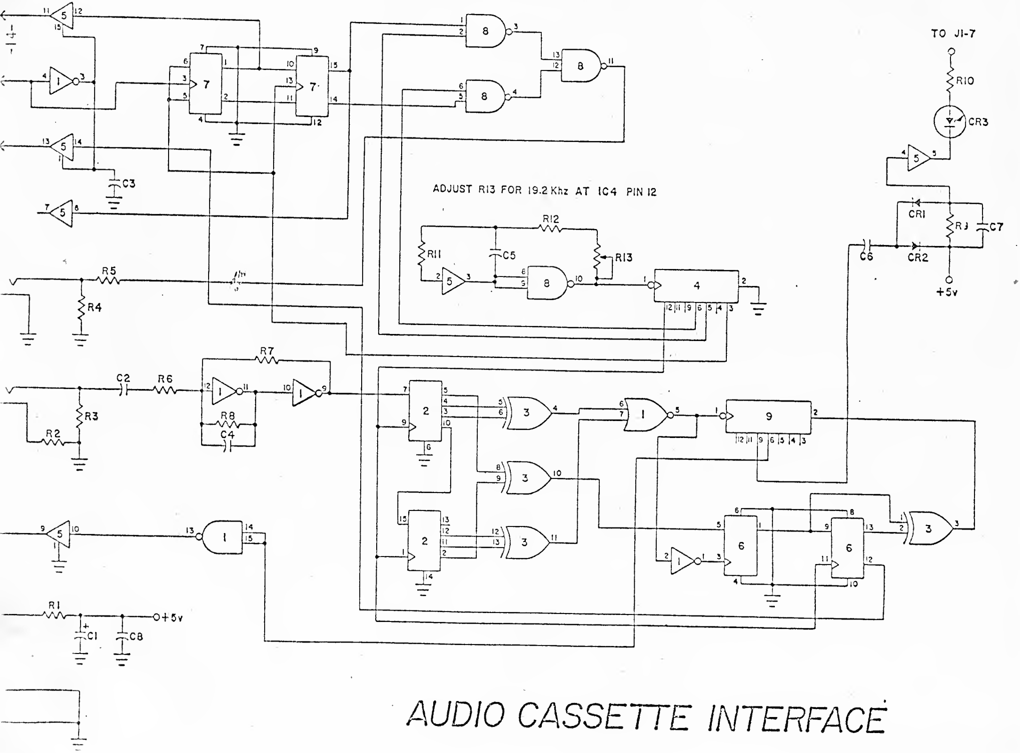
The programmer must deal with this inconvenience by combining these distinct values together using an arithmetic expression. One easy way to do this is to multiply the higher order or leftmost value by the power of 2 of its rightmost bit position and then add in the other (right side) value.

CHARACTER CODE TABLE

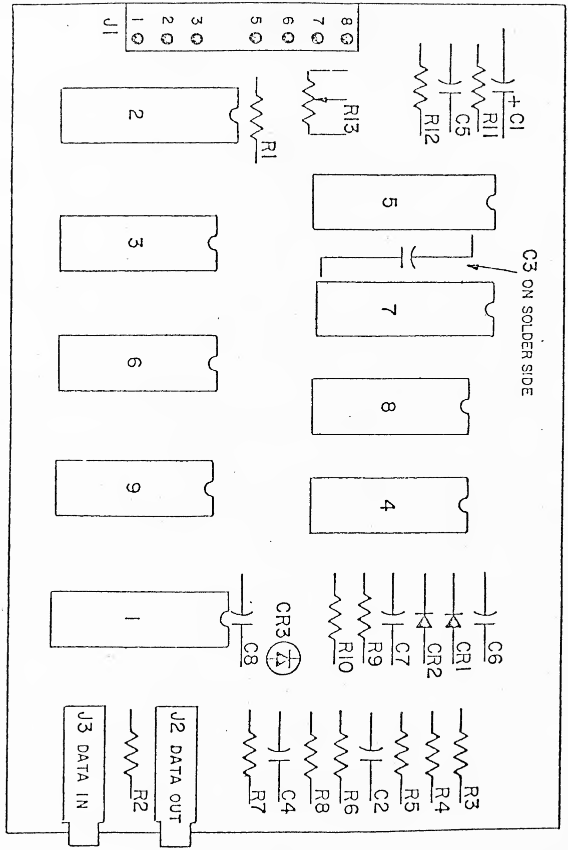
#	CHARACTER	#	CHARACTER	#	CHARACTER	#	CHARACTER
0	??	32	SPACE	64	@	96	↓
1	??	33	!	65	A	97	→
2	??	34	"	66	B	98	X(Multiply)
3	??	35	#	67	C	99	÷
4	??	36	$	68	D	100	??
5	??	37	%	69	E	101	??
6	??	38	&	70	F	102	??
7	??	39	'(apostrophe)	71	G	103	??
8	??	40	(	72	H	104	LIST
9	??	41	)	73	I	105	CLEAR
10	??	42	*	74	J	106	RUN
11	??	43	+	75	K	107	NEXT
12	??	44	,	76	L	108	LINE
13	GO	45	-	77	M	109	IF
14	??	46	. (period)	78	N	110	GOTO
15	??	47	/	79	O	111	GOSUB
16	??	48	0	80	P	112	RETURN
17	??	49	1	81	Q	113	BOX
18	??	50	2	82	R	114	FOR
19	??	51	3	83	S	115	INPUT
20	??	52	4	84	T	116	PRINT
21	??	53	5	85	U	117	STEP
22	??	54	6	86	V	118	RND
23	??	55	7	87	W	119	TO
24	??	56	8	88	X	120	??
25	??	57	9	89	Y	121	??
26	??	58	:	90	Z	122	??
27	??	59	:	91	[	123	??
28	??	60		92	\	124	??
29	??	61	=	93	]	125	??
30	??	62		94	↑	126	??
31	ERASE	63	?	95	←	127	??

MEMORY AREAS OF INTEREST

	DECIMAL	HEXIDECIMAL
ON BOARD ROM-	0-8191	0-1FFF
BALLY BASIC ROM-	8192-12287	2000-2FFF
SCREEN MEMORY AREA-	16384-20479	4000-4FFF
BALLY BASIC GRAPHICS/-PROGRAM AREA-	16384-19983	4ØØØ-4E1Ø
BALLY BASIC SCRATCHPAD MEMORY AREA -	20000-20463	4E2Ø-4FEF
TAPE INPUT BUFFER -	20002-20049	4E22-4E51
VARIABLES BEGIN AT -	20078	4E6E
LINE INPUT BUFFER (104 characters) -	20180-20283	4ED4-4F3B
STACK AREA -	20284-20462	4F3C-4FEE
TEXT AREA -	24576-22777	ØA000÷ØA707
NOTE LOOKUP TABLE	12046	2FØE for CR($13_{1Ø}$)



AUDIO CASSETTE INTERFACE


J1
1
2
3
5
6
7
8
+ C1
R11
C5
R12
R13
R1
2
5
3
C3 ON SOLDER SIDE
7
6
8
9
4
1
C8
CR3
R10
R9
C7
CR2
CR1
C6
J3 DATA IN
R2
J2 DATA OUT
R7
C4
R8
R6
C2
R5
R4
R3

R1 - 270 ohm	C1 - 22mfd 6v Tantalum
R2 - 100 ohm	C2 - .1 mfd
R3 - 47 ohm	C3 - 100 pfd
R4 - 100 ohm	C4 - 100 pfd
R5 - 51K ohm	C5 - 100 pfd
R6 - 10K ohm	C6 - 470 pfd
R7 - 1M ohm	C7 - 470 pfd
R8 - 100K ohm	C8 - .1 mfd bypass
R9 - 10M ohm	IC1 - MC14572UB
R10- 270 ohm	IC2 - MC14015B
R11- 330K ohm	IC3 - MC14070B
R12- 150K ohm	IC4 - MC14024B
ALL ABOVE 1/4W 5%	IC5 - MC14503B
R13- 20K ohm trimpot	IC6 - MC14013B
	IC7 - MC14027B
CR1 - 1N4148	IC8 - MC14011B
CR2 - 1N4148	IC9 - MC14024B
CR3 - MV5754 LED	